Package 'CINdex'

April 15, 2020

Title Chromosome Instability Index

Version 1.14.0

Description The CINdex package addresses important area of high-throughput genomic analysis. It allows the automated processing and analysis of the experimental DNA copy number data generated by Affymetrix SNP 6.0 arrays or similar high throughput technologies. It calculates the chromosome instability (CIN) index that allows to quantitatively characterize genome-wide DNA copy number alterations as a measure of chromosomal instability. This package calculates not only overall genomic instability, but also instability in terms of copy number gains and losses separately at the chromosome and cytoband level.

Depends R (>= 3.3), GenomicRanges

License GPL (>= 2)

LazyData true

Imports bitops,gplots,grDevices,som, dplyr,gridExtra,png,stringr,S4Vectors, IRanges, GenomeInfoDb,graphics, stats, utils

biocViews Software, CopyNumberVariation, GenomicVariation, aCGH, Microarray, Genetics, Sequencing

Suggests knitr, testthat, ReactomePA, RUnit, BiocGenerics, AnnotationHub, rtracklayer, pd.genomewidesnp.6, org.Hs.eg.db, biovizBase, TxDb.Hsapiens.UCSC.hg18.knownGene, methods, Biostrings,Homo.sapiens

VignetteBuilder knitr

NeedsCompilation no

Author Lei Song, Krithika Bhuvaneshwar, Yue Wang, Yuanjian Feng, Ie-Ming Shih, Subha Madhavan, Yuriy Gusev

Maintainer Yuriy Gusev <yg63@georgetown.edu>

RoxygenNote 5.0.1

git_url https://git.bioconductor.org/packages/CINdex

git_branch RELEASE_3_10

git_last_commit 0ae2a81

git_last_commit_date 2019-10-29

Date/Publication 2020-04-14

R topics documented:

`clin.crc` *Colon cancer clinical dataset*

Description

The example dataset consisits of 10 colon cancer patients, of which 5 had relapse (return of cancer to colon) and the rest did not relapse. This example dataset is part of the complete dataset used in *CRC*, and can be accessed via G-DOC Plus at https://gdoc.georgetown.edu. The column names are described below:

Usage

```
data(clin.crc)
```

Format

A matrix with 10 rows and 2 columns

Details

- Sample. Sample ID
- Label. Refers to the group label/outcome

More details on how this object was created is provided in the vignette titled "How to prepare Input data" in the CINdex package.

Value

An example clinical dataset

cnvgr.18.auto — *Probe annotation file for Affymetrix Genome Wide Human SNP Array 6.0*

Description

This is a probe annotation file for Affymetrix Genome Wide Human SNP Array 6.0. It contains annotation for only the copy number probes in this array and corresponds to hg18 reference genome.

The GRanges object contains details about probe name, chromosome number, start end position and strand. The annotation has been filtered to include only those probes that are located in autosomes.

More details on how this object was created is provided in the vignette titled "How to prepare Input data" in the CINdex package.

Usage

```
data(cnvgr.18.auto)
```

Format

A GRanges object

Value

An example probe annotation file

comp.heatmap — *A comprehensive heatmap function that plots Chromosome and Cytoband heatmaps*

Description

When the `run.cin.chr` and `run.cyto.chr` functions are called, we get Chromosome and Cytoband CIN values for various gain/loss threshold settings. This `comp.heatmap` function can be used to pick the best threshold for the input data. It plots heatmaps for two groups of interest (case and control) for all the input gain/loss threshold settings. By visually checking the heatmaps, the user can pick the threshold/setting that shows the best contrast between two groups of interest. Steps: #Step 1: Run cytoband CIN or chromosome CIN - using `run.cin.chr()` or `run.cin.cyto()` #Step 2: Call this function to create chromosome or cytoband level heatmaps. Pick gain/loss threshold appropriate for data. See vignette for more details.

Usage

```
comp.heatmap(R_or_C = "Regular", clinical.inf = NULL, genome.ucsc = NULL,
  in.folder.name = "output_chr_cin", out.folder.name = "output_chr_plots",
  plot.choice = "png", base.color = "black", thr.gain = c(2.5, 2.25, 2.1),
  thr.loss = c(1.5, 1.75, 1.9), V.def = 2:3, V.mode = c("sum", "amp",
  "del"))
```

Arguments

R_or_C	The value'Regular' plots chromosome level heatmap and 'Cytobands' plots cytoband level heatmaps
clinical.inf	An n*2 matrix, the 1st column is 'sample name', the second is 'label'
genome.ucsc	A Reference genome
in.folder.name	Name of folder where the Chromsome CIN or Cytoband CIN objects are present
out.folder.name	Name of folder where the Chromosome heatmaps or Cytoband heatmaps will be saved
plot.choice	A choice of whether the heatmaps should be .png or .pdf format
base.color	A choice of 'black' or 'white' base color for the heatmap (indicating no instability)
thr.gain	A threshold above which will be set as gain
thr.loss	A threshold below which will be set as loss
V.def	There are 2 different CIN definitions - normalized (value=2) and un-normalized (value=3)
V.mode	There are 3 options: 'sum', 'amp' and 'del'

Value

No value returned. If R_or_C='Regular', it will genearte chromosome level heatmap, If R_or_C='Cytobands',it will generate cytoband level heatmap

See Also

See accompanying vignette for end-to-end tutorial

Examples

```
###### Example 1 - Chromosome level

## Step 1: Run chromosome CIN
# This is how command should be run:
## Not run:
run.cin.chr(grl.seg = grl.data)

## End(Not run)
# For this example, we run chr CIN on one threshold only
data("grl.data")
run.cin.chr(grl.seg = grl.data, thr.gain=2.25, thr.loss=1.75, V.def=3, V.mode="sum")

## Step 2: Plot chromosome level heatmap
# This is how the command must be called:
## Not run:
comp.heatmap(R_or_C="Regular", clinical.inf=clin.crc, genome.ucsc=hg18.ucsctrack, thr.gain = 2.25,
thr.loss = 1.75,V.def = 3,V.mode = "sum")

## End(Not run)
# For this example, we run chr heatmap on one threshold only
comp.heatmap(R_or_C='Regular', clinical.inf=clin.crc, genome.ucsc=hg18.ucsctrack, thr.gain = 2.25,
thr.loss = 1.75,V.def = 3,V.mode = "sum")
```

```
###### Example 2 - Cytoband level

## Step 1 : Run cytoband CIN
# This is how command should be run:
## Not run:
run.cin.cyto(grl.seg = grl.data,cnvgr=cnvgr.18.auto, snpgr=snpgr.18.auto,
genome.ucsc = hg18.ucsctrack)

## Step 2: Plot cytoband level heatmap

comp.heatmap(R_or_C="Cytobands", clinical.inf=clin.crc, genome.ucsc=hg18.ucsctrack,
thr.gain=2.25, thr.loss=1.75,V.def=3,V.mode="sum")

## End(Not run)
```

`cyto.cin4heatmap` *Cytoband CIN T-test output*

Description

Example output obtained from running the T-test on Cytoband CIN object. See accompanying vignette in the CINdex package for a complete tutorial

Usage

```
data(cyto.cin4heatmap)
```

Format

List

Value

Cytoband CIN T-test output

`cytobands.cin` *Cytoband CIN dataset*

Description

Example output obtained from running the Cytoband CIN function in the CINdex package. Indicates chromsome instability index value for every cytoband.

Usage

```
data(cytobands.cin)
```

Format

List

Value

An example cytoband CIN

`extract.genes.in.cyto.regions` *Given an input of cytobands, it outputs a list of genes that are present in the cytoband regions*

Description

Once the user has a list of cytobands of interest, one downstream application could be to find the list of genes present in the cytoband regions. This `extract.genes.in.cyto.regions` function can be used for this purpose. The following steps should be run before this function can be called: #Step 1 : Run cytoband CIN - using `run.cin.chr()` #Step 2: Plot cytoband level heatmap - using `comp.heatmap()` #Step 3: Go through heatmaps as select one appropriate threshold. Load the file. #Step 4: Perform T test to find differentially expressed cytobands - using `ttest.cyto.cin.heatmap()` #Step 5: Call this funtion to extract genes located in cytoband regions #More details and tutorial are given in the accompanying vignette

Usage

```
extract.genes.in.cyto.regions(cyto.cin4heatmapObj = NULL,
  genome.ucsc = NULL, gene.annotations = NULL,
  folder.name = "output_genename")
```

Arguments

`cyto.cin4heatmapObj`	Output of the cytoband T test results
`genome.ucsc`	Reference sequence
`gene.annotations`	Information about CDS start and end positions, Gene names
`folder.name`	Name of output folder

Value

Output files: The genes names present in the cytoband regions

See Also

See accompanying vignette for an end-to-end tutorial

Examples

```
#For this example, we load example T test output object
data("cyto.cin4heatmap")
data("hg18.ucsctrack") #load Hg 18 reference annotation file
data("geneAnno") #load Gene annotations file
extract.genes.in.cyto.regions(cyto.cin4heatmapObj =cyto.cin4heatmap,
genome.ucsc = hg18.ucsctrack, gene.annotations = geneAnno)
```

geneAnno *CDS gene annotation file*

Description

A CDS gene annotation file with the following column names (obtained for human reference)

- chrom. Chromosome number
- strand. Positive or negative strand
- cdsStart. CDS Start position
- cdsEnd. CDS end position
- GeneID. Gene symbol

More details on how this object was created is provided in the vignette titled "How to prepare Input data" in the CINdex package.

Usage

```
data(geneAnno)
```

Format

A matrix

Value

An example CDS gene annotation file

grl.data *Output of segmentation algorithm*

Description

To mathematically and quantitatively describe these alternations we first locate their genomic positions and measure their ranges. Such algorithms are referred to as segmentation algorithms. Bioconductor has several copy number segmentation algorithms. There are many copy number segmentation algorithms outside of Bioconductor as well, examples are Fused Margin Regression (FMR) and Circular Binary Segmentation (CBS).

Segmentation results are typically have information about the start position and end position in the genome, and the segment value. The algorithms typically covers chromosomes 1 to 22 without any gaps, sometimes sex chromosomes are also included.

For more details refer tutorial in the accompanying vignette in the CINdex package

Usage

```
data(grl.data)
```

Format

A GRangesList

Value

An example output of segmentation algorithm

`hg18.ucsctrack` *Human reference annotation file*

Description

The reference annotation file used in the CIN algorithm. The example file used here is for Human Species hg18 and includes information about chromosome number, start and end position, name of cytoband and stain.

More details on how this object was created is provided in the vignette titled "How to prepare Input data" in the CINdex package.

Usage

```
data(hg18.ucsctrack)
```

Format

GRanges object

Value

An example hg18 annotation file

`run.cin.chr` *Calculate chromosome CIN*

Description

`run.cin.chr` calculates chromosome level CIN for the following default thresholds (with and without normalization): (a) gain threshold 2.5 and loss threshold 1.5 (b) gain threshold 2.25 and loss threshold 1.75 (c) gain threshold 2.10 and loss threshold 1.90. For each of these threshold settings, this function will calculate CIN for gains, losses, and a combination of gains and losses (referred to as 'sum' or 'overall' CIN). This will allow user to examine and select the best setting of gain and loss threshold for their data. More details and tutorial are given in the accompanying vignette.

Usage

```
run.cin.chr(grl.seg, out.folder.name = "output_chr_cin", thr.gain = c(2.5,
  2.25, 2.1), thr.loss = c(1.5, 1.75, 1.9), V.def = 2:3, V.mode = c("sum",
  "amp", "del"))
```

Arguments

grl.seg	The result of any segmentation algorithm such as CBS,FMR. Should be a data frame of 3 column-lists or matrix of three-column lists
out.folder.name	Name of output folder, where the CIN ojbects for each setting will be created
thr.gain	A numeric list that contains values set as threshold gain
thr.loss	A numeric list that contains values set as threshold loss
V.def	An integer vector that has different CIN definitions (2 means normalized, 3 means un-normalized)
V.mode	A vector that has 3 options: 'sum', 'amp' and 'del'

Value

Creates a dataMatrix R object for each setting that contains CIN values

See Also

See accompanying vignette for end-to-end tutorial

Examples

```
# Run chromosome level CIN calculation for all thresholds. This is how command should be run:
# A number of RData objects will be created in 'output_chr' folder.
## Not run:
run.cin.chr(grl.seg = grl.data)

## End(Not run)

#For this example, we run this function for one threshold only

data("grl.data")
run.cin.chr(grl.seg = grl.data, thr.gain=2.25, thr.loss=1.75, V.def=3, V.mode="sum")

# Next step: Plot chromosome level heatmap \code{\link{comp.heatmap}}
# More details and tutorial are given in the accompanying vignette
```

---

run.cin.cyto *Calculate cytoband CIN*

Description

run.cyto.chr calculates cytoband level CIN for the following default thresholds (with and without normalization): (a) gain threshold 2.5 and loss threshold 1.5 (b) gain threshold 2.25 and loss threshold 1.75 (c) gain threshold 2.10 and loss threshold 1.90. For each of these threshold settings, this function will calculate CIN for gains, losses, and a combination of gains and losses (referred to as 'sum' or 'overall' CIN). This will allow user to examine and select the best setting of gain and loss threshold for their data. More details and tutorial are given in the accompanying vignette.

Usage

```
run.cin.cyto(grl.seg, cnvgr = NULL, snpgr = NULL, genome.ucsc,
  out.folder.name = "output_cyto_cin", thr.gain = c(2.5, 2.25, 2.1),
  thr.loss = c(1.5, 1.75, 1.9), V.def = 2:3, V.mode = c("sum", "amp",
  "del"), chr.num = 22)
```

Arguments

grl.seg	The result of any segmentation algorithm such as CBS,FMR. Should be a GRanges-List
cnvgr	Probe annotation info for the copy number probes - GRanges object
snpgr	Probe annotation info for the SNP probes - GRanges object
genome.ucsc	A Reference genome
out.folder.name	Name of output folder, where the CIN objects for each setting will be created
thr.gain	A numeric list that contains values set as threshold gain
thr.loss	A numeric list that contains values set as threshold loss
V.def	An integer vector that has 2 different CIN definitions - normalized (value=2) and un-normalized (value=3)
V.mode	A vector that has 3 options: 'sum', 'amp' and 'del'
chr.num	Number of chromosomes in input. Typically 22.

Value

Creates a dataMatrix and cytobands.cin R objects for each setting that contains CIN values

See Also

Accompanying vignette for complete end-to-end tutorial

Examples

```
#### For this example, we run cytoband CIN calculation for one setting on chromosome 1 only
data("grl.data") #need segment level data

#getting genome reference file
data("hg18.ucsctrack")
hg18.ucsctrack.chr <- subset(hg18.ucsctrack, seqnames(hg18.ucsctrack) %in% "chr22")

#get probe annotation information
data("cnvgr.18.auto")

#Call function to run cytoband CIN
run.cin.cyto(grl.seg = grl.data, cnvgr=cnvgr.18.auto, snpgr=NULL,
genome.ucsc = hg18.ucsctrack.chr, thr.gain = 2.25,thr.loss = 1.75,
V.def = 3, V.mode="sum",chr.num = 22)

#Run cytoband level CIN calculation for all thresholds. This is how command should be run:
## Not run:
run.cin.cyto(grl.seg = grl.data, cnvgr=cnvgr.18.auto, snpgr=snpgr.18.auto,
genome.ucsc = hg18.ucsctrack)
```

```
## End(Not run)
# A number of RData objects will be created in 'output_cyto' folder.
```

---

snpgr.18.auto *Probe annotation file for Affymetrix Genome Wide Human SNP Array 6.0*

Description

This is a probe annotation file for Affymetrix Genome Wide Human SNP Array 6.0. It contains annotation for only the SNP probes in this array and corresponds to hg18 reference genome.

The GRanges object contains details about probe name, chromosome number, physical location and strand. The annotation has been filtered to include only those probes that are located in autosomes.

More details on how this object was created is provided in the vignette titled "How to prepare Input data" in the CINdex package.

Usage

```
data(snpgr.18.auto)
```

Format

A GRanges object

Value

An example probe annotation file

---

ttest.cyto.cin.heatmap *Performs T test on cytoband level CIN data, and plots heatmap*

Description

ttest.cyto.cin.heatmap to perform T test to find differentially expressed cytobands. It also plots a heatmap after performing heirarchical clustering. When to use this function: #Step 1: Run cytoband CIN - using run.cin.chr(). #Step 2: Plot cytoband level heatmap - using comp.heatmap(). #Step 3: Go through heatmaps as select one appropriate threshold. Load the file. #Step 4: Call this function. More details and tutorial are given in the accompanying vignette

Usage

```
ttest.cyto.cin.heatmap(cytobands.cin.obj, clinical.inf, genome.ucsc,
  file.ext = "gainT_lossT_unnorm", folder.name = "output_ttest",
  combine.cyto.flag = FALSE)
```

Arguments

cytobands.cin.obj	(eg. cytobands.cin_2.25_1.75_unnormalized_amp.Rdata), a list in which each cell is chromosome cin matrix
clinical.inf	In a clinical.inf.Rdata is a two columns array, the 1st column is samplename, the 2nd is the label
genome.ucsc	Reference sequence
file.ext	Provide a meaningful file name extension. Ideally include the gain, loss threshold settings
folder.name	Name of folder where the output files will be generated
combine.cyto.flag	Whether or not to save the combine cytobands as a uni array rather than a list

Value

#Outputs: 1. cyto.cin.uni.file.ext.Rdata (eg. cyto.cin.uni.gainT_lossT_unnormalized.Rdata) 2. Heatmaps:
eg. CIN relapse-free VS relapse for gainT_lossT_unnormalized_dendrogram.pdf 3. Raw CIN array
for the corresponding heatmap: #ttest.cyto.cin4heatmap.gainT_lossT_unnormalized.csv #ttest.cyto.cin4heatmap.gainT_l
4. T test results for all cytobands on the whole genome #ttest.cytobands.cin.gainT_lossT_unnormalized.txt

See Also

See accompaying vignette for a detailed end to end workflow tutorial

Examples

```
#For this example, we load an example cytoband CIN data
data("cytobands.cin")
data("clin.crc") # sample names with group information
data("hg18.ucsctrack") #hg18 reference file
ttest.cyto.cin.heatmap(cytobands.cin.obj = cytobands.cin,
clinical.inf = clin.crc, genome.ucsc = hg18.ucsctrack)
```

Index